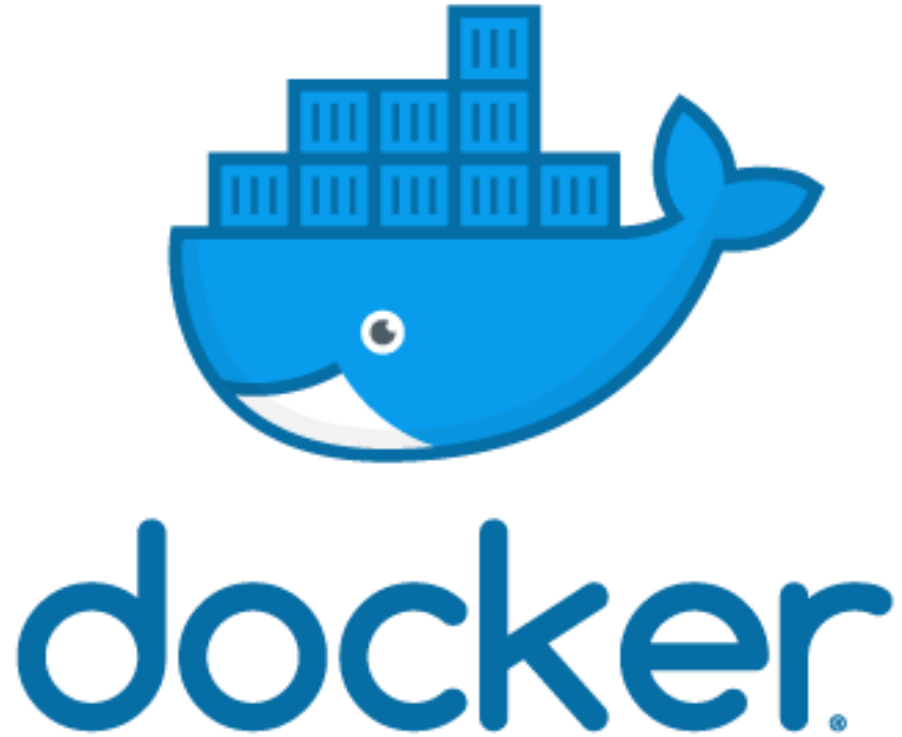
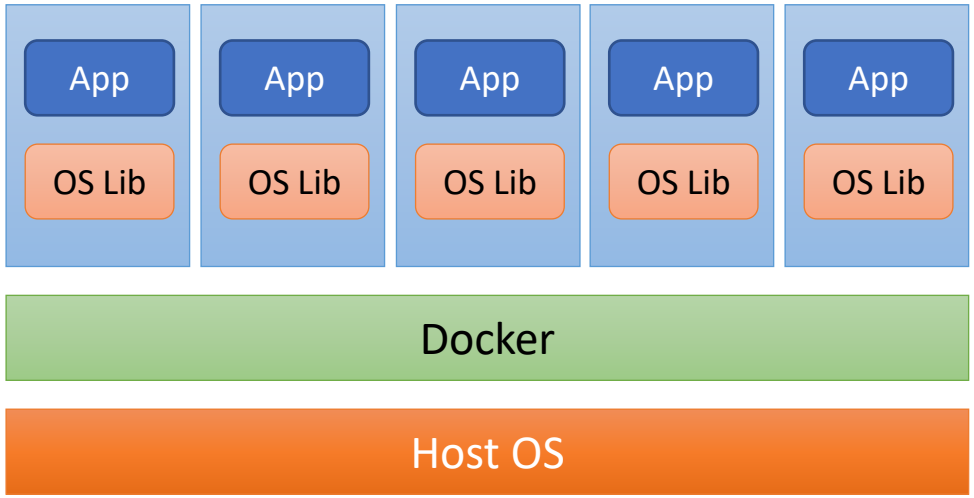
Quick Docker Tips

by

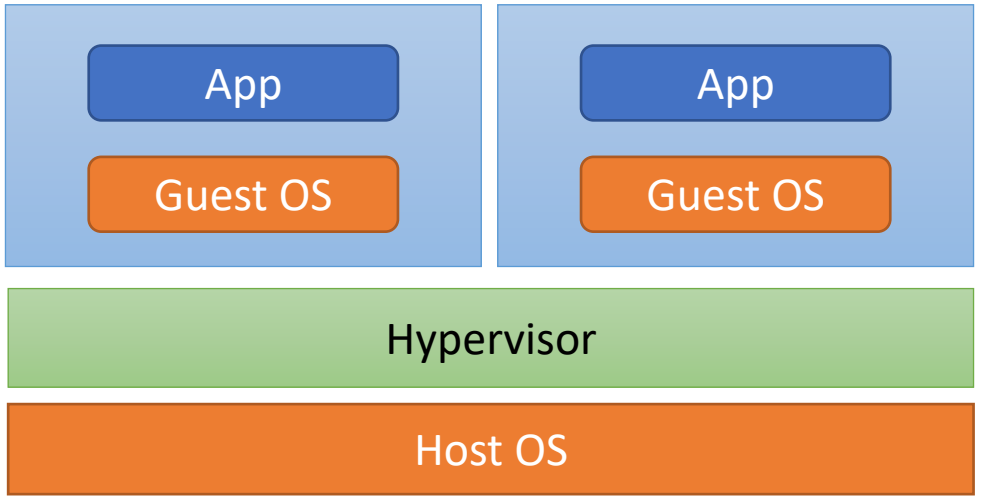
Marian Veteanu



Docker: OS level virtualization



Hypervisor: Hardware level virtualization



- OS level virtualization
- Very low overhead
- Containers more like processes

Display Docker information

$ docker version

$ docker info

Display existing Docker images

$ docker images

Display running containers

$ docker ps

Display all containers (including stopped containers)

$ docker ps -a

Stopping a container

$ docker stop [cid]

Removing a stopped container

$ docker rm [cid]

Removing all stopped containers

$ docker container prune

Removing an existing image

$ docker rmi [cid]

Download a container image

$ docker pull ubuntu

Quick Docker tips by
Marian Veteanu

Running a container image

```
$ docker run --rm -it ubuntu bash
```

```
# run ubuntu image
# -- rm removes the created container on exit
# launch the bash process in interactive mode (-it)
```

Download and run a container image containing a server process

```
$ docker run --name testnginx -d -p 8080:80 nginx
```

```
$ docker inspect testnginx
```

```
# download and run the nginx image (https://hub.docker.com/_/nginx/)
# --name sets a name to the created containers
# -d disconnects
# -p maps port 8080 of local machine to port 80 of container
# in inspect output verify the IP address of the container
```

Mounting an external folder (volume) inside a container image

```
$ docker run --rm -it -v $PWD:/src ubuntu bash
$ docker run --rm -it -v %CD%:/src ubuntu bash
```

mount current folder ($PWD, %CD%) in folder /src of the docker image

Use node.js from inside docker image to run .js files on host drive

```
$ docker run --rm -it --name node -v $PWD:/src -w /src node bash
$ docker run --rm -it --name node -v %CD%:/src -w /src node bash
```

```
$ cd src
$ node t.js
```

use node or other dev tools without installing on local machine
use different versions of the same tools (e.g. different images for different node versions)

Use a node.js container to run a web server application

Create w.js file (example found google-ing "node.js web server")

```
var http = require('http');

//create a server object:
http.createServer(function (req, res) {
  res.write('Hello World!'); //write a response to the client
  res.end(); //end the response
}).listen(8080); //the server object listens on port 8080
```

$ docker run --rm -p 8080:8080 -it --name node -d -v %CD%:/src -w /src node node w.js

Create a container image with a node.js application

- Place w.js in folder js
- Create file Dockerfile

```
FROM node
RUN mkdir -p /js
COPY ./js/* /js
WORKDIR /js
CMD node w.js
```

$ docker build . -t testnode
$ docker run --rm -p 8080:8080 -d testnode

See also online example for aci-helloworld
https://docs.microsoft.com/en-us/azure/container-instances/container-instances-tutorial-prepare-app

Create a container image with a go application

- Run exercise on Linux
- Create t1.go (from the first page of golang tutorial)

```
package main

import "fmt"
import "os"

func main() {
        fmt.Println("Hello from the world of GoLang!", os.Getenv("DICTIONARY"))
}
```

- Create Dockerfile

```
FROM alpine:3.5
RUN mkdir -p /usr/p
COPY ./t1 /usr/p
WORKDIR /usr/p
CMD /usr/p/t1
```

$ go build t1.go
$ docker build . -t vmago
$ docker run -e DICTIONARY="MY DB CONNECTION" vmago

Docker compose demo

- Create docker-compose.yml

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - dbdata:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
volumes:
    dbdata:
```

Start the Docker compose environment

$ docker-compose up -d

Stop the Docker compose environment

$ docker-compose down --volumes

Useful docker images - Portainer

- Info at https://portainer.readthedocs.io/en/latest/deployment.html

$ docker run -d -p 9000:9000 --restart always -v /var/run/docker.sock:/var/run/docker.sock -v /opt/portainer:/data portainer/portainer

Useful docker images - Azure CLI

- Also available at http://shell.azure.com

$ docker run -it microsoft/azure-cli

Example of Azure CLI commands:

$ az container create --name nginx --image nginx --ip-address public -g vmamachines

$ az container list -o table

Windows containers

- Windows Server Core
- Nano Server
(and variants: aspnet)

$ docker pull microsoft/aspnet

Dockerize an ASP.NET application

- Create ASP.NET web application in Visual Studio 2017
- Publish and then run in publish folder: 'dotnet WebApplication1.dll'
- Create Dockerfile

```
FROM microsoft/aspnetcore
WORKDIR /app
COPY . .
ENTRYPOINT ["dotnet", "WebApplication1.dll"]
```

$ docker build -t myasp .
$ docker run --rm -d --name myaspcontainer myasp
$ docker inspect myaspcontainer

Running commands on a running container

$ docker exec [cid] ipconfig

$ docker exec -it [cid] cmd

Run the above commands against a running Windows container

Marian Veteanu

Technology Architect and Product Leader

Looking to see how I can add value to your organization? Message me!

https://www.linkedin.com/in/mveteanu/
https://x.com/mveteanu